# Week 2 Notes

## *Lectures 3 & 4*

*University of Massachusetts, Amherst*, CS389

# 1 Neural Networks (lecture 3)

**Definition 1.1** (Multi-dimensional data)**.** A *multi-dimensional data* has $n$ number of data and $d$ number of features and can be written as[1]:

$$X = \left[X^{(1)}, X^{(2)}, ..., X^{(n)}\right]^T \tag{1.1}$$

where $X^{(i)}$ is the $i^{\text{th}}$ data[2]. A single data, $X^{(i)}$ is a d-dimensional vector which has $d$ number of features. For example, the features for a weather data may include humidity, temperature, air speed, etc. Every data is represented as a row and each feature as a column. Formally we write a single data as:

$$X^{(i)} = \left[X_1^{(i)}, X_2^{(i)}, ..., X_d^{(i)}\right] \tag{1.2}$$

Putting all this together, our multi dimensional data is a matrix with dimensions $(n \times d)$:

$$\begin{bmatrix} X_1^{(1)} & ... & X_d^{(1)} \\ ... & ... & ... \\ X_1^{(n)} & ... & X_d^{(n)} \end{bmatrix} \tag{1.3}$$

## 1.1 Perceptron

**Definition 1.2** (Perceptron)**.**

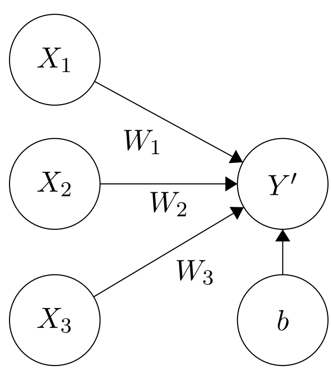


Figure 1: A single-layer perceptron

where the vertices $X_i$ are the inputs, the edges $W_i$ are the weighted connections, the vertex $b$ is the bias term, and the vertex $Y'$ is the output. We can be write this mathematically as:

$$Y' = \phi(W^T X + b) \tag{1.4}$$

where $\phi$ is a non-linearity function, also known as an activation function (topic of next week).

[1]Note that this array is transposed because $X$ is a row vector and each $X^{(i)}$ is a d-dimensional column vector

[2]The reason why we use the $X^{(i)}$ notation is because $X^{(i)}$ is a vector and sometimes we need to refer to the elements of that vector [2]

### 1.2 Classification

In a classification problem, we want to learn to predict discrete classes which the input belongs to.

**Definition 1.3** (Binary classification)**.** Is a supervised learning algorithm that *categorizes the data into one of two classes.*

Recall that a linear model is defined as:

$$\hat{y} = \phi\left(\sum_{j=1}^{m} w_j \cdot X_j + b\right) = \phi(W^T X + b) \tag{1.5}$$

where the output $\hat{y} \in \mathbb{R}$ and we can apply a boolean function (i.e. the *sign* activation function) so that the output $\hat{y} \in \{-1, +1\}$. Given a fixed model, the binary classifier is defined as:

$$\hat{y} = \text{sign}(W^T X + b) = \begin{cases} +1 & \text{if } W^T X + b > 0 \\ -1 & \text{otherwise} \end{cases} \tag{1.6}$$

As can be seen in Figure 2, the points that are on the same side as the normal vector to the hyperplane is the positive half-spaces are classified as positive. Likewise, the other half-space is negative and all points are classified as negative [2].

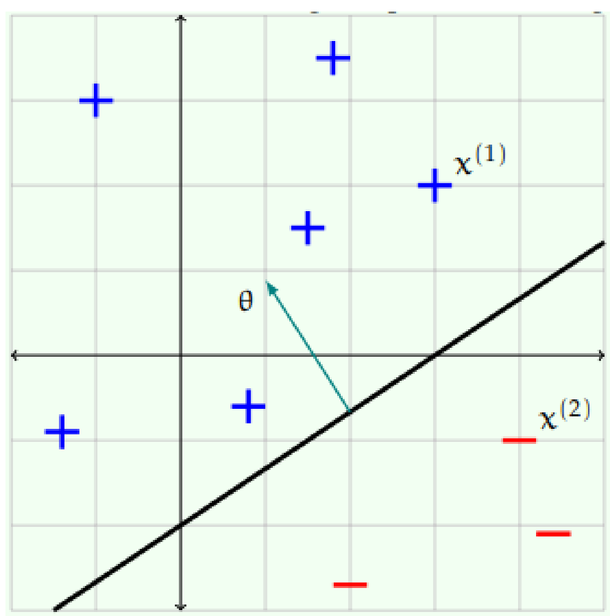


Figure 2: Binary Classification Example. The $\theta$ is the same variable as $W$ [2]

### 1.3 Linear Separability

**Theorem 1.** *If the dataset, $D$, is linearly separable, then the perceptron algorithm is guaranteed to find a linear separator* [2]

How would one *formally describe linear separabilty?* This is beyond the scope for this class, but you can refer to this MIT lecture notes for a detailed explanation. The intuition is if the shortest distance of a point to the hyperplane (the norm) is positive for all points then the dataset is classifed correctly. You can use Figure 2 to help with this visualization.

### 1.4 Regression

**Definition 1.4** (Regression)**.** Predict continous outputs ($\hat{y} \in \mathbb{R}$) that are close to the true values

## 2 Stochastic Gradient Descent implementation (lecture 4)

Hopefully, we now have a mathematical understanding and an intuition of all the main components of a supervised machine learning model. We can now start implementing a very simple SGD. Recall that in a SGD, we want to update the parameters $W$ and $b$ after every single training data.

**Algorithm 1** Stochastic Gradient Descent algorithm

**for** $e$ in range Epoch **do** ▷ number of epochs
    **for** $X^{(i)}$ in $X$ **do** ▷ loop through entire dataset
        $\hat{y}^{(i)} = F_W(X^{(i)}) = X^{(i)}W^T + b$ ▷ our model's prediction for input $X^{(i)}$
        $\text{Loss}^{(i)} = \text{MSE}(y^{(i)}, \hat{y}^{(i)}) = \frac{1}{2}\sum(y^{(i)} - \hat{y}^{(i)})^2$ ▷ calculate loss of prediction with actual
        $W = W - \alpha\nabla_W \text{Loss}^{(i)}$ ▷ update weight
        $b = b - \alpha\nabla_b \text{Loss}^{(i)}$ ▷ update bias
    **end for**
**end for**

### 2.1 Calculating Gradient of Loss

We define the gradient of loss as $\nabla_W \text{Loss}$, which can also be written as $\frac{\partial L}{\partial W}$, such that:

$$\nabla_W \text{Loss} = \frac{\partial L}{\partial W} = \left(\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}, ..., \frac{\partial L}{\partial W_m}\right) \tag{2.1}$$

Using the chain rule we have:

$$\frac{\partial L}{\partial W_i} = \frac{\partial \hat{y}}{\partial W_i}\frac{\partial L}{\partial \hat{y}} \tag{2.2}$$

Now recall that $\hat{y} = W^T X + b$ and $L = \frac{1}{2}\sum(y^{(i)} - \hat{y}^{(i)})^2$ , therefore:

$$\frac{d\hat{y}}{dW_1} = X_1 \tag{2.3}$$

$$\frac{d\hat{y}}{db} = 1 \tag{2.4}$$

## References


[1] Cooper.

[2] MIT Open Learning Library, 6.036, Spring 2020, https://openlearninglibrary.mit.edu/assets/courseware/v1/2481f8f2964716032b134db99e369b81/asset-v1:MITx+6.036+1T2019+type@asset+block/notes_chapter_Introduction.pdf